Complete Mediation

Sean Barnum, Cigital, Inc. [vita[3]]
Michael Gegick, Cigital, Inc. [vita[4]]

2005-09-12 L4 / D/P, L[5]

A software system that requires access checks to an object each time a subject requests access, especially for security-critical objects, decreases the chances of mistakenly giving elevated permissions to that subject. A system that checks the subject's permissions to an object only once can invite attackers to exploit that system. If the access control rights of a subject are decreased after the first time the rights are granted and the system does not check the next access to that object, then a permissions violation can occur. Caching permissions can increase the performance of a system, but at the cost of allowing secured objects to be accessed.

Detailed Description Excerpts

From Saltzer and Schroeder [Saltzer 75], Section: "Basic Principles of Information Protection" on page 9:

> Complete mediation: Every access to every object must be checked for authority. This principle, when systematically applied, is the primary underpinning of the protection system. It forces a system-wide view of access control, which in addition to normal operation includes initialization, recovery, shutdown, and maintenance. It implies that a foolproof method of identifying the source of every request must be devised. It also requires that proposals to gain performance by remembering the result of an authority check be examined skeptically. If a change in authority occurs, such remembered results must be systematically updated.

From Bishop [Bishop 03], Chapter 13: "Design Principles," Section 13.2.4: "Principle of Complete Mediation," pages 345-346:[9]

> This principle restricts the caching of information. This often leads to simpler implementations of mechanisms.
>
> Definition 13-4.The principle of complete mediation requires that all accesses to objects be checked to ensure they are allowed.
>
> Whenever a subject attempts to read an object, the operating system should mediate the action. First, it determines if the subject can read the object. If so, it provides the resources for the read to occur. If the subject tries to read the object again, the system should again check that the subject can still read the object. Most systems would not make the second check. They would cache the results of the first check, and base the second access upon the cached results.
>
> **Example 1**
>
> When a UNIX process tries to read a file, the operating system determines if the process is allowed to read the file. If so, the process receives a file descriptor encoding the allowed access. Whenever the process wants to read the file, it presents the file descriptor to the kernel. The kernel then allows the access. If the owner of the file disallows the process permission to read the file after the file descriptor is issued, the kernel still allows access. This scheme violates the principle of complete mediation, because the second access is not checked. The cached value is used, resulting in the denial of access being ineffective.
>
> **Example 2**

3. http://buildsecurityin.us-cert.gov/bsi/about_us/authors/35-BSI.html (Barnum, Sean)
4. http://buildsecurityin.us-cert.gov/bsi/about_us/authors/345-BSI.html (Gegick, Michael)
9. All rights reserved. It is reprinted with permission from Addison-Wesley Professional.

The Directory Name Service (DNS) caches information mapping hostnames into IP addresses. If an attacker is able to "poison" the cache by implanting records associating a bogus IP address with a name, the host will route connections to that host incorrectly. Section 14.6.1.2 [of *Computer Security: Art and Science*] discusses this in more detail.

Further Reading

We discuss race conditions and canonicalization as examples where a simple check on an object's permissions or name may not be adequate when performing a sensitive operation.

Race Conditions that Can Circumvent Access Control Checks

Even though multiple access checks may be implemented in their source code, developers should be warned that their security concerns are not necessarily over. Attackers can make use of race conditions that occur between an access check and the subsequent privileged operation. This type of race condition is called time of check, time of use (TOCTOU) and has been exploited in the UNIX operating system. The following code belongs to a program running setuid root [Viega 02]:

```
/* access returns 0 on success */
if (!access(file, W_OK)) {
  f = fopen(file, "wb+");
  write_to_file(f);
} else {
  fprintf(stderr, "Permission denied when trying to open %s. \n", file);
}
```

Users can request that this program write to the files they own. The program first checks to determine whether the real UID has permissions for the request and, if so, returns 0. In the time between when access privileges are checked and when the file is opened, an attacker can replace the file he or she has rights to with a file that is owned by root. An attacker can perform the swap by creating a dummy file (e.g., /etc/passwd) with his or her permission and then creating a symbolic link to it. The attacker can then perform a command such as [Viega 02]

```
$ rm pointer; ln -s /etc/passwd pointer
```

It is then possible that the program will overwrite the system password file [Viega 02]. Therefore, developers should be cautioned to make sure their access checks cannot be subverted with race conditions such as those in the class of TOCTOU.

Careful with Canonicalization

Upon checking a resource for permissions, the name of that resource should correctly identify the object you expect to check. Different names can be used to apply to one object. For example, a file can be named c:\dir\trip.gif, trip.gif, and ..\..\trip.gif. The process of resolving various equivalent forms of a filename into a standard name is called *canonicalization*. The final (or canonical) name in this example could be c:\dir\trip.gif [Howard 02].

An example of where filenames were represented differently to subvert security mechanisms was in the Napster application. Napster implemented filters to restrict selected songs from being shared, in accordance with a federal judge's ruling. Users began to change the filenames of the songs that were ordered not to be shared. For example, one could possibly translate song titles into leet-speak (e.g., Cold Play's "White Shadows"to "VV]-[!73 5|-|4|)0VV2" or Audioslave's "Be Yourself" into "|33 ?/()|_||2531|^}-{"). The filtering mechanisms for those songs did not adequately address the canonicalization of restricted song titles [Howard 02].

References

[Bishop 03]	Bishop, Matt. *Computer Security: Art and Science*. Boston, MA: Addison-Wesley, 2003.
[Howard 02]	Howard, Michael & LeBlanc, David. *Writing Secure Code, 2nd ed*. Redmond, WA: Microsoft Press, 2002.
[Saltzer 75]	Saltzer, Jerome H. & Schroeder, Michael D. "The Protection of Information in Computer Systems." 1278-1308. *Proceedings of the IEEE 63,* 9 (September 1975).
[Viega 02]	Viega, John & McGraw, Gary. *Building Secure Software: How to Avoid Security Problems the Right Way*. Boston, MA: Addison-Wesley, 2002.

Cigital, Inc. Copyright

1. mailto:copyright@cigital.com